University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

ECE 120: Introduction to Computing

Letter Frequency Coding

Review the Problem to Be Solved

The task:

- given an **ASCII** string (terminated by **NUL**)
- count the occurrences of each letter (regardless of case), and
- the number of non-alphabetic characters.

The high-level approach:

initialize histogram to all 0s
for each character in the string
increment the appropriate histogram bin

Where Are the Pieces in Memory?

Let's start with some notes about
where we want to store information

x4000	the start of the string
x3000	the start of our code
x3100	non-alpha histogram bin
x3101 to **x311A**	alpha bins A to Z (in order)

What Shall We Keep in the Registers?

For the counting part, we will
use registers as follows

R0	histogram pointer (x3100)
R1	string pointer (moves)
R2	current character from string
R3, **R4**, **R5**	ASCII constants (to be chosen)
R6	temporary

Get a Pointer to the Histogram into R0

x3000 LEA R0,xFF

We could use an LD instruction.

But there's a better way, without storing x3100 in memory.

We need to initialize R0 to x3100.

We Also Need to Fill the Histogram with 0s

The next step: fill the histogram with 0s.

We need registers.

Let's reuse a few (so far, only **R0** is initialized).

- **R1** a loop counter (27 iterations)
- **R2** current histogram bin to fill
- **R6** the number 0 (to store)

Prepare Our Registers to Initialize the Histogram

x3000 LEA R0,xFF
x3001 AND R6,R6,#0

To set R6 to 0, use an AND.

Now, we need to initialize R6 to 0, R1 to #27, and R2 to x3100.

Prepare Our Registers to Initialize the Histogram

x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______

What about R1?

Let's just store #27 somewhere and use an LD.

Now, we need to initialize R6 to 0, R1 to #27, and R2 to x3100.

Prepare Our Registers to Initialize the Histogram

x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______
x3003 ADD R2,R0,#0

Now, we need to initialize R6 to 0, R1 to #27, and R2 to x3100.

And what about R2?

Remember that R0 already has the value x3100!

We're Ready to Fill the Histogram with 0s

Remember our register contents:

- **R1** a loop counter (27 iterations)
- **R2** current histogram bin to fill
- **R6** the number 0 (to store)

In our loop body, we write one 0 (from **R6**) to a bin at the memory location pointed to by **R2**.

Then we point to the next bin (increment **R2**).

Then we decrement our loop counter (**R1**).

Finally, we loop until the counter reaches 0.

Fill One Histogram Bin with 0

x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______
x3003 ADD R2,R0,#0
x3004 STR R6,R2,#0

Write one 0 (from R6) to the histogram bin to which R2 points.

Is there an LC-3 instruction for that?

Point to the Next Histogram Bin

x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______
x3003 ADD R2,R0,#0
x3004 STR R6,R2,#0
x3005 ADD R2,R2,#1

Point R2 to the next bin.

Is there an LC-3 instruction for that?

Decrement the Loop Counter

```
x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______
x3003 ADD R2,R0,#0
x3004 STR R6,R2,#0
x3005 ADD R2,R2,#1
x3006 ADD R1,R1,#-1
```

Decrement the loop counter.

Is there an LC-3 instruction for that?

Branch Backward Until We Finish Filling the Histogram

```
x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______
x3003 ADD R2,R0,#0
x3004 STR R6,R2,#0
x3005 ADD R2,R2,#1
x3006 ADD R1,R1,#-1
x3007 BRp #-4
```

Branch backward until we have written 27 bins.

Is there an LC-3 instruction for that?

R1 started at #27.

Memory Addresses Do Not Appear in Real Code

```
x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______
x3003 ADD R2,R0,#0
x3004 STR R6,R2,#0
x3005 ADD R2,R2,#1
x3006 ADD R1,R1,#-1
x3007 BRp #-4
```

Now the histogram is filled with 0s.

See the memory addresses?

Those are just for us. They're not really in the code, as you should already know.

In Binary Programs, Instructions Must Appear as Bits

```
x3000 LEA R0,xFF
x3001 AND R6,R6,#0
x3002 LD R1, ______
x3003 ADD R2,R0,#0
x3004 STR R6,R2,#0
x3005 ADD R2,R2,#1
x3006 ADD R1,R1,#-1
x3007 BRp #-4
```

Now the histogram is filled with 0s.

See the instructions?

So far, those are more like our comments. Soon, you can write code that way.

We Still Have Initialization Work to Do

What about these other registers?

R1	string pointer (moves)
R2	current character from string
R3, **R4**, **R5**	ASCII constants (to be chosen)
R6	temporary

Let's initialize them now.
(No need to initialize **R2** nor **R6**.)

Initialize the Remaining Registers with LD

```
x3008 LD R3,x1B
x3009 LD R4,x1B
x300A LD R5,x1B
x300B LD R1,x1B
```

Initialize the other registers using LD.

Look good?

Are those all loading the same value?

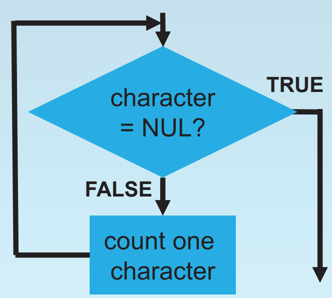
The addresses are PC-relative, so each loads from a separate memory location!

Ready to Count Letters?

Now we are finally ready to count letters!

Before We Can Count, We Must Load a Character

The first step?
- Load a character from the string, and
- check if it's **NUL**.



Load a Character from the String

x300C LDR R2,R1,#0

Load a character from the string.

Remember that R1 points to the next character in the string.

Also remember that we want the character in R2.

If We Find a NUL, We are Done

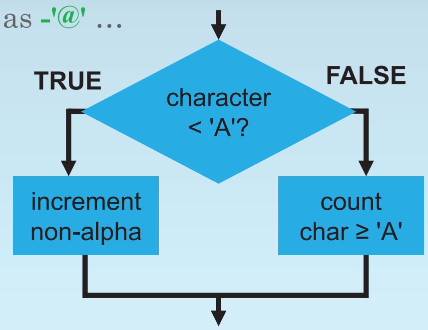
x300C LDR R2,R1,#0
x300D BRz ______

Check for NUL (x00).

Now We Can Classify the Character

We need to compare with capital **A**.

Let's define **R3** as **-'@'** …



Subtract @ to Compare with Capital A

Remember the ASCII table?

x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

Subtracting **'@'** allows us to check for non-alphabetic characters in the left region.

We store the difference (original character minus **'@'**) back in **R2**, so A through Z become 1 through 26.

Subtract @ to Compare with Capital A

```
x300C LDR R2,R1,#0
x300D BRz ______
x300E ADD R2,R2,R3
```

Compare with capital A.

Add R3 (-'@') to R2 and write the sum back into R2.

Branch Unless We Have a Character in the Left Region

```
x300C LDR R2,R1,#0
x300D BRz ______
x300E ADD R2,R2,R3
x300F BRp ______
```

Branch forward if the character is not in the left non-alphabetic region.

What is the branch condition?

Time to Increment the Non-Alpha Histogram Bin

x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

If the result is not positive,
- the character is in the left region and
- is not a letter.

So we can increment the non-alpha bin (at x3100).

Increment Memory Location x3100 (Non-Alpha Bin)

```
x300C LDR R2,R1,#0
x300D BRz ______
x300E ADD R2,R2,R3
x300F BRp ______
x3010 LDR R6,R0,#0
```

Increment memory at x3100 (the value held in R0).

So … ?

Is there an LC-3 instruction for that?

Where should we put the value?

No.

Increment Memory Location x3100 (Non-Alpha Bin)

```
x300C LDR R2,R1,#0
x300D BRz ______
x300E ADD R2,R2,R3
x300F BRp ______
x3010 LDR R6,R0,#0
x3011 ADD R6,R6,#1
```

Increment memory at x3100 (the value held in R0).

And now increment the value.

Increment Memory Location x3100 (Non-Alpha Bin)

```
x300C LDR R2,R1,#0
x300D BRz ______
x300E ADD R2,R2,R3
x300F BRp ______
x3010 LDR R6,R0,#0
x3011 ADD R6,R6,#1
x3012 STR R6,R0,#0
```

Increment memory at x3100 (the value held in R0).

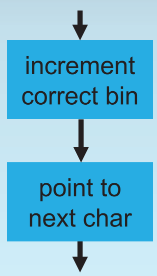
And put the new value back.

We Are Done with That Character

We are done counting that character.

The loop is inside the first task shown here (the one labeled "increment correct bin").

So now we need to **point to the next character…**



Go to the End of the Loop

```
x300C LDR R2,R1,#0
x300D BRz ______
x300E ADD R2,R2,R3
x300F BRp __x4__
x3010 LDR R6,R0,#0
x3011 ADD R6,R6,#1
x3012 STR R6,R0,#0
x3013 BRnzp ______
```

(somewhere)

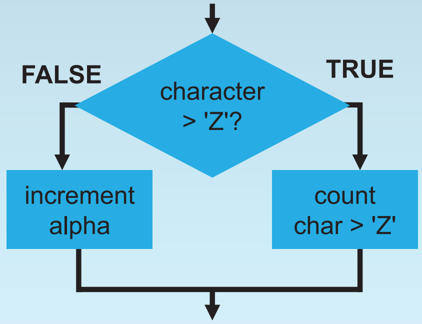
We are done counting this character.

Branch (always) to the end of our loop.

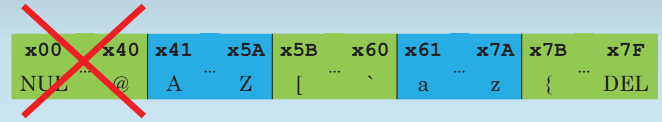
We can fill this offset in now.

We Need to Check for a Capital Letter

Next, we compare with capital **Z**.



Subtract Z to Make the Next Comparison



This time, we want to subtract **'Z'**.

But we already subtracted **'@'**, so now we add **'@' - 'Z'** (let's keep this value in **R4**).

We discard the result (store the result in **R6**).

Add (@ – Z) to Compare with Capital Z

x3014 ADD R6,R2,R4

Compare with capital Z.

Add R4 ('@' – 'Z') to R2 and write the sum into R6.

Branch Unless We Have a Capital Letter

x3014 ADD R6,R2,R4
x3015 BRp ______

Branch forward if the character is not a capital letter.

What is the branch condition?

Remember: we just calculated (original character – 'Z')

Time to Increment the One Letter's Histogram Bin

x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

If the result is not positive,
the character is a capital letter.

What bin should we increment?

(Hint: R2 now holds 1 to 26 for A to Z.)

The bin at address x3100 + R2.

Increment One Letter's Histogram Bin

```
x3014 ADD R6,R2,R4
x3015 BRp ______
x3016 ADD R2,R2,R0
```

Increment memory at x3100 + R2 (R0 + R2).

Where can we put the bin pointer?

First, we need to calculate a bin pointer.

We only need R2 to find the right bin.

Increment One Letter's Histogram Bin

```
x3014 ADD R6,R2,R4
x3015 BRp ______
x3016 ADD R2,R2,R0
x3017 LDR R6,R2,#0
```

Increment memory at address pointed to by R2.

Is there an LC-3 instruction for that?

Same answer as last time: load, modify, store.

Increment One Letter's Histogram Bin

```
x3014 ADD R6,R2,R4
x3015 BRp ______
x3016 ADD R2,R2,R0
x3017 LDR R6,R2,#0
x3018 ADD R6,R6,#1
```

Increment memory at address pointed to by R2.

And now increment the value.

Increment One Letter's Histogram Bin

x3014 ADD R6,R2,R4
x3015 BRp ______
x3016 ADD R2,R2,R0
x3017 LDR R6,R2,#0
x3018 ADD R6,R6,#1
x3019 STR R6,R2,#0

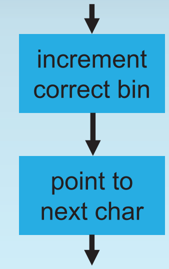
Increment memory at address pointed to by R2.

And put the new value back.

We Are Done with That Character

As before, we are done with that character.

So now we need to **point to the next character…**



Go to the End of the Loop

x3014 ADD R6,R2,R4
x3015 BRp __x5__
x3016 ADD R2,R2,R0
x3017 LDR R6,R2,#0
x3018 ADD R6,R6,#1
x3019 STR R6,R2,#0
x301A BRnzp ______

(somewhere)

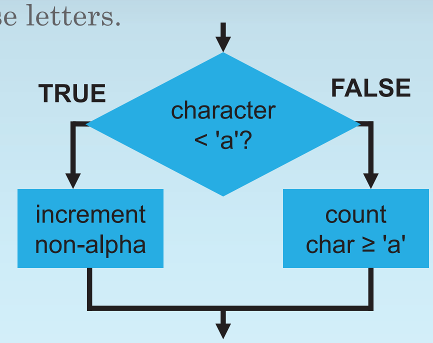
We are done counting this character.

Branch (always) to the end of our loop.

We can fill this offset in now.

We Need to Check for the Middle Region

Next, we want to look for the start of the lower case letters.



Subtract x60 to Make the Next Comparison

x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

We want to subtract **x60** (**backquote**, **'`'**).

But we already subtracted **'@'** from **R2**, so now add **'@' - '`'** (let's keep this value in **R5**).

Let's write the result back to **R2** so that lower case letters produce values 1 to 26 in **R2**.

Add (@ – `) to Compare with Lower Case a

x301B ADD R2,R2,R5

Compare with lower case a.

Add R5 ('@' – '`') to R2 and write the sum back to R2.

When Do We Have a Character in the Middle Region?

x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

We just wrote (**original character minus x60**) into **R2**.

Under what conditions (N, Z, P) do we have a character in the middle region?

N and Z

How Can We Increment the Non-Alpha Bin?

x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

So for conditions **N** or **Z**, we want to increment the non-alpha bin.

How?

Didn't we already write that code?

Let's just branch to it!

Branch If We Have a Character in the Middle Region

x301B ADD R2,R2,R5
x301C BRnz ______

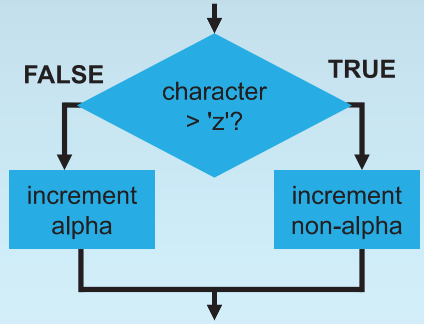
Handle characters in the middle region.

So what is the branch condition?

We can find the right offset now (the code is already written), but let's just leave it for later.

We Need to Check for a Lower Case Letter

Next, we compare with lower case **z**.



Subtract z to Make the Next Comparison

x00 … x40	x41 … x5A	x5B … x60	x61 … x7A	x7B … x7F
NUL … @	A … Z	[… `	a … z	{ … DEL

This time, we want to subtract **'z'**.

But we already subtracted **'`'**, so now we add **'`' - 'z'** (it's already in **R4**!).

We discard the result (store the result in **R6**).

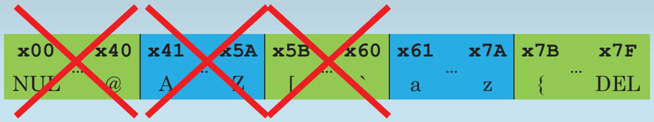
Add (` – z) to Compare with Lower Case z

x301B ADD R2,R2,R5
x301C BRnz ______
x301D ADD R6,R2,R4

Compare with lower case z.

Add R4 ('`' – 'z') to R2 and write the sum into R6.

When Do We Have a Lower Case Letter?

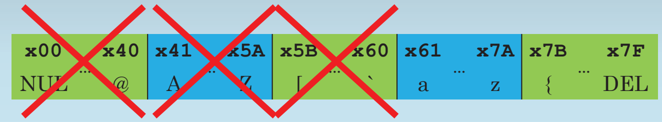


We just wrote (**original character minus 'z'**) into **R6**.

Under what conditions (N, Z, P) do we have a lower case letter?

N and Z

How Can We Increment the Right Letter's Bin?



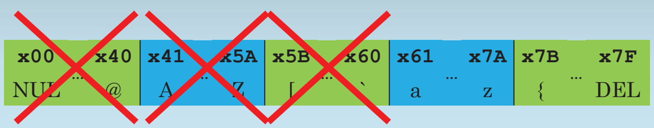
So for conditions **N** or **Z**, we want to increment one of the letter's histogram bins.

How?

Didn't we already write that code?

Let's just branch to it!

How Can We Increment the Right Letter's Bin?



Let's be clear:
- We are able to reuse the code because **we designed the code to be reusable**.
- In both cases, **R0** points to the histogram, and **R2** is 1 to 26 for the letter.

Branch If We Have a Lower Case Letter

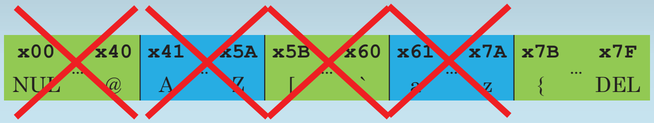
```
x301B ADD R2,R2,R5
x301C BRnz ______
x301D ADD R6,R2,R4
x301E BRnz ______
```

Handle lower case letters.

What is the branch condition?

Again, we can find the right offset, but we'll just leave it blank.

We Know that the Character is Not a Letter



At this point, we know that the original character was not a letter.

So … ?

Branch (unconditionally) to the code that increments the non-alpha histogram bin.

Branch to the Code for Non-Alphabetic Characters

```
x301B ADD R2,R2,R5
x301C BRnz ______
x301D ADD R6,R2,R4
x301E BRnz ______
x301F BRnzp ______
```

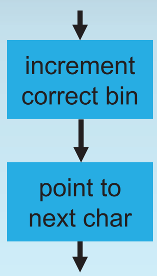
Handle the last region.

Again, we can find the right offset, but we'll just leave it blank.

Next, Advance the String Pointer

We are now finished with the upper task.

We can write the code to point to the next character.



Advance the String Pointer to the Next Character

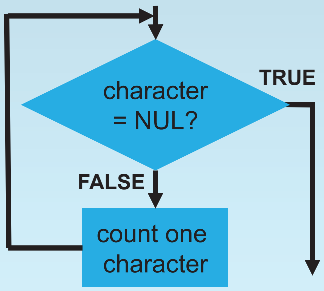
```
x301B ADD R2,R2,R5
x301C BRnz ______
x301D ADD R6,R2,R4
x301E BRnz ______
x301F BRnzp ______
x3020 ADD R1,R1,#1
```

Advance the string pointer (in R1).

Is there an LC-3 instruction for that?

Our Loop Body is Complete

And now we're done with counting a character and advancing the string pointer, so we can return to the start of our loop.



Return to the Start of the Loop

x301B ADD R2,R2,R5
x301C BRnz ______
x301D ADD R6,R2,R4
x301E BRnz ______
x301F BRnzp ______
x3020 ADD R1,R1,#1
x3021 BRnzp ______

Return to the start of the loop.

Is there an LC-3 instruction for that?

We Need a HALT and Some Data

x3022 TRAP x25
x3023 #27
x3024 -'@'
x3025 '@' – 'Z'
x3026 '@' – '`'
x3027 x4000

We need a HALT and some data.

The full program is available online, both as a printout and as real code.

Be sure to write a string before you run the code (unless you like 0s).

The Rest is Left to You

I'll leave the rest for you.

All of the counting.
All of the bits.
All of the fun, really!

